Exercises 3: Linear smoothing and Gaussian processes

Basic concepts

Bias–variance decomposition

Let $\hat{f}(x)$ be a noisy estimate of some function $f(x)$, evaluated at some point x. Define the mean-squared error of the estimate as

$$\mathrm{MSE}(\hat{f}, f) = \mathrm{E}\{[f(x) - \hat{f}(x)]^2\}\,.$$

A simple derivation shows that $\mathrm{MSE}(f, \hat{f}) = B^2 + v$, where

$$B = \mathrm{E}\{\hat{f}(x)\} - f(x) \quad \text{and} \quad V = \mathrm{var}\{\hat{f}(x)\}\,.$$

variance of $\hat{f}$'s estimator

when $B = 0$: we get a unbiased estimator

This is the bias–variance decomposition of mean-squared error.

A simple example (optional problem)

Some people refer to the above decomposition as the *bias–variance tradeoff*. Why a tradeoff? Here's a simple example to convey the intuition.

Suppose we observe $x_1, \ldots, x_n$ from some distribution F, and want to estimate $f(0)$, the value of the probability density function at 0. Let h be a small positive number, called the *bandwidth*, and define the quantity

$$\pi_h = P\left(-\frac{h}{2} < X < \frac{h}{2}\right) = \int_{-h/2}^{h/2} f(x)dx\,.$$

Clearly $\pi_h \approx hf(0)$. idea of Riemann sum: pdf in a small neighborhood is approximately the same

(A) Let Y be the number of observations in a sample of size n that fall within the interval $(-h/2, h/2)$. What is the distribution of Y? What are its mean and variance in terms of n and π_h? Propose a simple estimator $\hat{f}(0)$ involving Y. $Y \sim$ Binomial (n, π_h)

$\mathrm{E}(Y) = n\pi_h$. Take m samples, each with size n; we let $\bar{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i = n\pi_h = nh\hat{f}(0) \Rightarrow \hat{f}(0) = \frac{\bar{Y}}{nh}$

(B) Suppose we expand $f(x)$ in a second-order Taylor series about 0:

$$f(x) \approx f(0) + xf'(0) + \frac{x^2}{2}f''(0)\,.$$

Use this in the above expression for π_h, together with the bias–variance decomposition, to show that

$$\mathrm{MSE}\{\hat{f}(0), f(0)\} \approx Ah^4 + \frac{B}{nh}$$

for constants A and B that you should (approximately) specify. What happens to the bias and variance when you make h small? When you make h big?

(C) Use this result to derive an expression for the bandwidth that minimizes mean-squared error, as a function of n. You can approximate any constants that appear, but make sure you get the right functional dependence on the sample size.

Curve fitting by linear smoothing

Consider a nonlinear regression problem with one predictor and one response: $y_i = f(x_i) + \epsilon_i$, where the ϵ_i are mean-zero random variables.

(A) Suppose we want to estimate the value of the regression function $y^\star$ at some new point $x^\star$, denoted $\hat{f}(x^\star)$. Assume for the moment that $f(x)$ is linear, and that y and x have already had their means subtracted, in which case $y_i = \beta x_i + \epsilon_i$.

Return to your least-squares estimator for multiple regression. Show that for the one-predictor case, your prediction $\hat{y}^\star = f(x^\star) = \hat{\beta} x^\star$ may be expressed as a *linear smoother*[1] of the following form:

$$\hat{f}(x^\star) = \sum_{i=1}^{n} w(x_i, x^\star) y_i$$

for any $x^\star$. Inspect the weighting function you derived. Briefly describe your understanding of how the resulting smoother behaves, compared with the smoother that arises from an alternate form of the weight function $w(x_i, x^\star)$:

$$w_K(x_i, x^\star) = \begin{cases} 1/K, & x_i \text{ one of the } K \text{ closest sample points to } x^\star, \\ 0, & \text{otherwise.} \end{cases}$$

This is referred to as *K-nearest-neighbor smoothing*.

[1] This doesn't mean the weight function $w(\cdot)$ is linear in its arguments, but rather than the new prediction is a linear combination of the past outcomes y_i.

$\hat{\beta} = \frac{\sum x_i y_i}{\sum x_i^2} \Rightarrow w(x_i, x^*) = \frac{x_i}{\sum x_i^2} x^*$

Linear combination of y_i's, each of which has its weight determined by the magnitude of product of corresponding independent variable x_i and new data x^*.

For KNN smoothing: eliminate affects from points far away — more conservative approach.

(B) A *kernel function* $K(x)$ is a smooth function satisyfing

$$\int_{\mathbb{R}} K(x)dx = 1\,, \quad \int_{\mathbb{R}} xK(x)dx = 0\,, \quad \int_{\mathbb{R}} x^2 K(x)dx > 0\,.$$

its a density $\quad E_K(x) = 0 \quad$ $\mathrm{Var}(x) = E_K(x^2) > 0$

A very simple example is the uniform kernel,

$$K(x) = \frac{1}{2} I(x) \quad \text{where} \quad I(x) = \begin{cases} 1, & |x| \le 1 \\ 0, & \text{otherwise}\,. \end{cases}$$

Another common example is the Gaussian kernel:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,.$$

Kernels are used as weighting functions for taking local averages. Specifically, define the weighting function

$$w(x_i, x^\star) = \frac{1}{h} K\left(\frac{x_i - x^\star}{h}\right),$$

where h is the bandwidth. Using this weighting function in a linear smoother is called *kernel regression*. (The weighting function gives the unnormalized weights; you should normalize the weights so that they sum to 1.)

Write your own R function that will fit a kernel smoother for an arbitrary set of x-y pairs, and arbitrary choice of (positive real) bandwidth h.[2] Set up an R script that will simulate noisy data from some nonlinear function, $y = f(x) + \epsilon$; subtract the sample means from the simulated x and y; and use your function to fit the kernel smoother for some choice of h. Plot the estimated functions for a range of bandwidths large enough to yield noticeable changes in the qualitative behavior of the prediction functions.

[2] Coding tip: write things in a modular way. A kernel function is a function accepting a distance and a bandwidth and returning a nonnegative real. A weighting function is a function that accepts a vector of previous x's, a new x, and a kernel function; and that returns a vector of weights. Et cetera. You will appreciate your foresight in writing modular code later in the course!

Cross validation

Left unanswered so far in our previous study of kernel regression is the question: how does one choose the bandwidth h used for the kernel? Assume for now that the goal is to predict well, not necessarily to recover the truth. (These are related but distinct goals.)

(A) Presumably a good choice of h would be one that led to smaller predictive errors on fresh data. Write a function or script that will: (1) accept an old ("training") data set and a new ("testing") data set as inputs; (2) fit the kernel-regression estimator to the training data for specified choices of h; and (3) return the estimated functions and the realized prediction error on the testing data for each value of h. This should involve a fairly straightforward "wrapper" of the function you've already written.

(B) Imagine a conceptual two-by-two table for the unknown, true state of affairs. The rows of the table are "wiggly function" and "smooth function," and the columns are "highly noisy observations" and "not so noisy observations." Simulate one data set (say, 500 points) for each of the four cells of this table, where the x's take values in the unit interval. Then split each data set into training and testing subsets. You choose the functions.[3] Apply your method to each case, using the testing data to select a bandwidth parameter.

[3] Trigonometric functions, for example, can be pretty wiggly if you make the period small.

Choose the estimate that minimizes the average squared error in prediction, which estimates the mean-squared error:

$$L_n(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n^\star}(y_i^\star - \hat{y}_i^\star)^2\,,$$

where $(y_i^\star, x_i^\star)$ are the points in the test set, and $\hat{y}_i^\star$ is your predicted value arising from the model you fit using only the training data. Does your out-of-sample predictive validation method lead to reasonable choices of h for each case?

(C) (optional problem) Splitting a data set into two chunks to choose h by out-of-sample validation has some drawbacks. List at least two. Then consider an alternative: leave-one-out cross validation. Define

$$\text{LOOCV} = \sum_{i=1}^{n}\left(y_i - \hat{y}_i^{(-i)}\right)^2\,,$$

where $\hat{y}_i^{(-i)}$ is the predicted value of y_i obtained by omitting the ith pair and fitting the model to the reduced data set.[4] This is contingent upon a particular bandwidth, and is obviously a function of x_i, but these dependencies are suppressed for notational ease. This looks expensive to compute: for each value of h, and for each data point to be held out, fit a whole nonlinear regression model. But you will derive a shortcut!

Observe that for a linear smoother, we can write the whole vector of fitted values as $\hat{y} = Hy$, where H is called the smoothing matrix (or "hat matrix") and y is the vector of observed outcomes.[5] Write $\hat{y}_i$ in terms of H and y, and show that $\hat{y}_i^{(-i)} = \hat{y}_i - H_{ii}y_i + H_{ii}\hat{y}_i^{(-i)}$. Deduce that, for a linear smoother,

$$\text{LOOCV} = \sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - H_{ii}}\right)^2\,.$$

[4] The intuition here is straightforward: for each possible choice of h, you have to predict each data point using all the others. The bandwidth that with the lowest prediction error is the "best" choice by the LOOCV criterion.

[5] Remember that in multiple linear regression this is also true:

$$\hat{y} = X\hat{\beta} = X(X^TX)^{-1}X^Ty = Hy\,.$$

Local polynomial regression

Kernel regression has a nice interpretation as a "locally constant" estimator, obtained from locally weighted least squares. To see this, suppose we observe pairs (x_i, y_i) for $i = 1, \ldots, n$ from our new favorite model, $y_i = f(x_i) + \epsilon_i$ and wish to estimate the value of the underlying function $f(x)$ at some point x by weighted least squares. Our estimate is the scalar[6] quantity

where is x in the following function?

$$\hat{f}(x) = a = \arg\min_{\mathbb{R}} \sum_{i=1}^{n} w_i(y_i - a)^2\,,$$

[6] Because we are only talking about the value of the function at a specific point x, not the whole function.

where the w_i are the normalized weights (i.e. they have been rescaled to sum to 1 for fixed x). Clearly if $w_i = 1/n$, the estimate is simply $\bar{y}$, the sample mean, which is the "best" globally constant estimator. Using elementary calculus, it is easy to see that if the unnormalized weights are

$$w_i \equiv w(x, x_i) = \frac{1}{h} K\left(\frac{x_i - x}{h}\right),$$

then the solution is exactly the kernel-regression estimator.

(A) A natural generalization of locally constant regression is local polynomial regression. For points u in a neighborhood of the target point x, define the polynomial

$$g_x(u; a) = a_0 + \sum_{k=1}^{D} a_j (u - x)^k$$

for some vector of coefficients $a = (a_0, \ldots, a_D)$. As above, we will estimate the coefficients a in $g_x(u; a)$ at some target point x using weighted least squares:

$$\hat{a} = \arg\min_{R^{D+1}} \sum_{i=1}^{n} w_i \{y_i - g_x(x_i; a)\}^2,$$

where $w_i \equiv w(x_i, x)$ are the kernel weights defined just above, normalized to sum to one.[7] Derive a concise (matrix) form of the weight vector $\hat{a}$, and by extension, the local function estimate $\hat{f}(x)$ at the target value x.[8] Life will be easier if you define the matrix R_x whose (i, j) entry is $(x_i - x)^{j-1}$, and remember that (weighted) polynomial regression is the same thing as (weighted) linear regression with a polynomial basis.

Note: if you get bogged down doing the general polynomial case, just try the linear case.

(B) From this, conclude that for the special case of the local linear estimator ($D = 1$), we can write $\hat{f}(x)$ as a linear smoother of the form

$$\hat{f}(x) = \frac{\sum_{i=1}^{n} w_i(x) y_i}{\sum_{i=1}^{n} w_i(x)},$$

where the unnormalized weights are

$$\begin{aligned} w_i(x) &= K\left(\frac{x - x_i}{h}\right) \{s_2(x) - (x_i - x) s_1(x)\} \\ s_j(x) &= \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) (x_i - x)^j. \end{aligned}$$

[7] We are fitting a different polynomial function for every possible choice of x. Thus $\hat{a}$ depends on the target point x, but we have suppressed this dependence for notational ease.

[8] Observe that at the target point x, $g_x(u = x; a) = a_0$. That is, only the constant term appears. But this is not the same thing as fitting only a constant term!

(C) Suppose that the residuals have constant variance σ^2 (that is, the spread of the residuals does not depend on x). Derive the mean and variance of the sampling distribution for the local polynomial estimate $\hat{f}(x)$ at some arbitrary point x. Note: the random variable $\hat{f}(x)$ is just a scalar quantity at x, not the whole function.

(D) We don't know the residual variance, but we can estimate it. A basic fact is that if x is a random vector with mean μ and covariance matrix Σ, then for any symmetric matrix Q of appropriate dimension, the quadratic form $x^T Q x$ has expectation

$$E(x^T Q x) = \text{tr}(Q\Sigma) + \mu^T Q \mu \, .$$

Write the vector of residuals as $r = y - \hat{y} = y - Hy$, where H is the smoothing matrix. Compute the expected value of the estimator

$$\hat{\sigma}^2 = \frac{\|r\|_2^2}{n - 2\text{tr}(H) + \text{tr}(H^T H)} \, ,$$

and simplify things as much as possible. Roughly under what circumstances will this estimator be nearly unbiased for large n? Note: the quantity $2\text{tr}(H) - \text{tr}(H^T H)$ is often referred to as the "effective degrees of freedom" in such problems.

"asymptotically diminishing bias".

(E) Write a new R function that fits the local linear estimator using a Gaussian kernel for a specified choice of bandwidth h. Then load the data in "utilities.csv" into R.[9] This data set shows the monthly gas bill (in dollars) for a single-family home in Minnesota, along with the average temperature in that month (in degrees F), and the number of billing days in that month. Let y be the average daily gas bill in a given month (i.e. dollars divided by billing days), and let x be the average temperature. Fit y versus x using local linear regression and some choice of kernel. Choose a bandwidth by leave-one-out cross-validation.

[9] On the class GitHub site.

(F) Inspect the residuals from the model you just fit. Does the assumption of constant variance (homoscedasticity) look reasonable? If not, do you have any suggestion for fixing it?

(G) Put everything together to construct an approximate point-wise 95% confidence interval for the local linear model (using your chosen bandwidth) for the value of the function at each of the observed points x_i for the utilities data. Plot these confidence bands, along with the estimated function, on top of a scatter plot of the data.[10]

[10] It's fine to use Gaussian critical values for your confidence set.

Gaussian processes

A *Gaussian process* is a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ such that, for any finite collection of indices $x_1, \ldots, x_N \in \mathcal{X}$, the random vector $[f(x_1), \ldots, f(x_N)]^T$ has a multivariate normal distribution. It is a generalization of the multivariate normal distribution to infinite-dimensional spaces. The set $\mathcal{X}$ is called the index set or the state space of the process, and need not be countable.

A Gaussian process can be thought of as a random function defined over $\mathcal{X}$, often the real line or $\mathbb{R}^p$. We write $f \sim \mathrm{GP}(m, C)$ for some mean function $m : \mathcal{X} \to \mathbb{R}$ and a covariance function $C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. These functions define the moments[11] of all finite-dimensional marginals of the process, in the sense that

$$E\{f(x_1)\} = m(x_1) \quad \text{and} \quad \mathrm{cov}\{f(x_1), f(x_2)\} = C(x_1, x_2)$$

for all $x_1, x_2 \in \mathcal{X}$. More generally, the random vector $[f(x_1), \ldots, f(x_N)]^T$ has covariance matrix whose (i, j) element is $C(x_i, x_j)$. Typical covariance functions are those that decay as a function of increasing distance between points x_1 and x_2. The notion is that $f(x_1)$ and $f(x_2)$ will have high covariance when x_1 and x_2 are close to each other.

[11] And therefore the entire distribution, because it is normal

(A) Read up on the Matern class[12] of covariance functions. The Matern class has the *squared exponential* covariance function as a special case:

$$C_{SE}(x_1, x_2) = \tau_1^2 \exp\left\{-\frac{1}{2}\left(\frac{d(x_1, x_2)}{b}\right)^2\right\} + \tau_2^2 \delta(x_1, x_2),$$

where $d(x_1, x_2) = \|x_1 - x_2\|_2$ is Euclidean distance (or just $|x - y|$ for scalars). The constants (b, τ_1^2, τ_2^2) are often called *hyperparameters*, and $\delta(a, b)$ is the Kronecker delta function that takes the value 1 if $a = b$, and 0 otherwise. But usually this covariance function generates functions that are "too smooth," and so we use other covariance functions in the Matern class as a default.[13]

Let's start with the simple case where $\mathcal{X} = [0, 1]$, the unit interval. Write a function that simulates a mean-zero Gaussian process on $[0, 1]$ under the Matern(5/2) covariance function. The function will accept as arguments: (1) finite set of points $x_1, \ldots, x_N$ on the unit interval; and (2) a triplet (b, τ_1^2, τ_2^2). It will return the value of the random process at each point: $f(x_1), \ldots, f(x_N)$.

Use your function to simulate (and plot) Gaussian processes across a range of values for b, τ_1^2, and τ_2^2. Try starting with a very small value of τ_2^2 (say, 10^{-6}) and playing around with the other two

[12] https://en.wikipedia.org/wiki/Matérn_covariance_function

[13] See the speed comparison in kernel-benchmark.R on the class GitHub site if you want to see how Rcpp can be used to speed things up here. My code is for the squared-exponential covariance function.

first. On the basis of your experiments, describe the role of these three hyperparameters in controlling the overall behavior of the random functions that result. What happens when you try $\tau_2^2 = 0$? Why? If you can fix this, do—remember our earlier discussion on different ways to simulate the MVN.

Now simulating a few functions with a different covariance function, the Matérn with parameter $5/2$:

$$C_{M52}(x_1, x_2) = \tau_1^2 \left\{ 1 + \frac{\sqrt{5}d}{b} + \frac{5d^2}{3b^2} \right\} \exp\left(\frac{-\sqrt{5}d}{b} \right) + \tau_2^2 \delta(x_1, x_2) \,,$$

where $d = \|x_1 - x_2\|_2$ is the distance between the two points x_1 and x_2. Comment on the differences between the functions generated from the two covariance kernels.[14]

[14] The Matérn covariance is actually a whole family of functions: http://en.wikipedia.org/wiki/MatŐrn_covariance_function.

(B) Suppose you observe the value of a Gaussian process $f \sim \mathrm{GP}(m, C)$ at points $x_1, \ldots, x_N$. What is the conditional distribution of the value of the process at some new point $x^\star$? For the sake of notational ease simply write the value of the (i, j) element of the covariance matrix as $C_{i,j}$, rather than expanding it in terms of a specific covariance function.

(C) Prove the following lemma.

Lemma 1. *Suppose that the joint distribution of two vectors y and θ has the following properties: (1) the conditional distribution for y given θ is multivariate normal, $(y \mid \theta) \sim N(R\theta, \Sigma)$; and (2) the marginal distribution of θ is multivariate normal, $\theta \sim N(m, V)$. Assume that R, Σ, m, and V are all constants. Then the joint distribution of y and θ is multivariate normal.*

In nonparametric regression and spatial smoothing

(A) Suppose we observe data $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, for some unknown function f. Suppose that the prior distribution for the unknown function is a mean-zero Gaussian process: $f \sim \mathrm{GP}(0, C)$ for some covariance function C. Let $x_1, \ldots, x_N$ denote the previously observed x points. Derive the posterior distribution for the random vector $[f(x_1), \ldots, f(x_N)]^T$, given the corresponding outcomes $y_1, \ldots, y_N$, assuming that you know σ^2.

(B) As before, suppose we observe data $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, for $i = 1, \ldots, N$. Now we wish to predict the value of the function $f(x^\star)$ at some new point $x^\star$ where we haven't seen previous data. Suppose that f has a mean-zero Gaussian process prior,

$f \sim GP(0, C)$. Show that the posterior mean $E\{f(x^\star) \mid y_1, \ldots, y_N\}$ is a linear smoother, and derive expressions both for the smoothing weights and the posterior variance of $f(x^\star)$.

(C) Go back to the utilities data, and plot the pointwise posterior mean and 95% posterior confidence interval for the value of the function at each of the observed points x_i (again, superimposed on top of the scatter plot of the data itself). Choose τ_2^2 to be very small, say 10^{-6}, and choose (b, τ_1^2) that give a sensible-looking answer.[15]

[15] If you're bored with the utilities data, instead try the data in ethanol.csv, in which the NOx emissions of an ethanol engine are measured as the engine's fuel-air equivalence ratio (E in the data set) is varied. Your goal would be to model NOx as a function of E using a Gaussian process.

(D) Let $y_i = f(x_i) + \epsilon_i$, and suppose that f has a Gaussian-process prior under the Matern(5/2) covariance function C with scale τ_2^1, range b, and nugget τ_2^2. Derive an expression for the marginal distribution of $y = (y_1 \ldots, y_N)$ in terms of (τ_1^2, b, τ_2^2), integrating out the random function f. This is called a marginal likelihood.

(E) Return to the utilities or ethanol data sets. Fix $\tau_2^2 = 0$, and evaluate the log of the marginal likelihood function $p(y \mid \tau_1^2, b)$ over a discrete 2-d grid of points.[16] If you're getting errors in your code with $\tau_2^2 = 0$, use something very small instead. Use this plot to choose a set of values $(\hat{\tau_1^2}, \hat{b})$ for the hyperparameters. Then use these hyperparameters to compute the posterior mean for f, given y. Comment on any lingering concerns you have with your fitted model.

[16] Don't just use a black-box optimizer; we want to make sure we get the best solution if there are multiple modes.

(F) In `weather.csv` you will find data on two variables from 147 weather stations in the American Pacific northwest.

pressure : the difference between the forecasted pressure and the actual pressure reading at that station (in Pascals)

temperature : the difference between the forecasted temperature and the actual temperature reading at that station (in Celsius)

There are also latitude and longitude coordinates of each station. Fit a Gaussian process model for each of the temperature and pressure variables. Choose hyperparameters appropriately. Visualize your fitted functions (both the posterior mean and posterior standard deviation) on a regular grid using something like a contour plot or color image. Read up on the `image`, `filled.contour`, or `contourplot`[17] functions in R. An important consideration: is Euclidean distance the appropriate measure to go into the covariance function? Or do we need separate length scales for the two dimensions, i.e.

$$d^2(x, z) = \frac{(x_1 - z_1)^2}{b_1^2} + \frac{(x_2 - z_2)^2}{b_2^2}.$$

Justify your reasoning for using Euclidean distance or this "non-isotropic" distance.

[17] in the lattice library